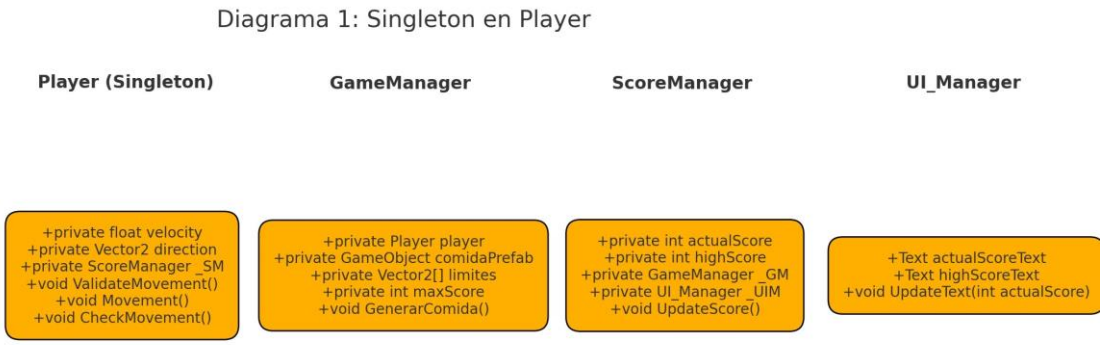

Diagrama 1

Seleccioné el Player como Singleton porque es crucial que solo exista una instancia del jugador durante todo el ciclo de vida del juego. En un juego como Snake, donde el Player es el personaje central, tener múltiples instancias no tendría sentido y podría causar problemas. Garantizar que solo haya un único Player simplifica el control y la interacción con otras clases.

La ventaja de esta elección es que centralizo el control del jugador. Esto facilita que otras clases, como el ScoreManager, accedan a las propiedades del jugador, como su posición o velocidad, de manera sencilla y sin necesidad de referencias complicadas.

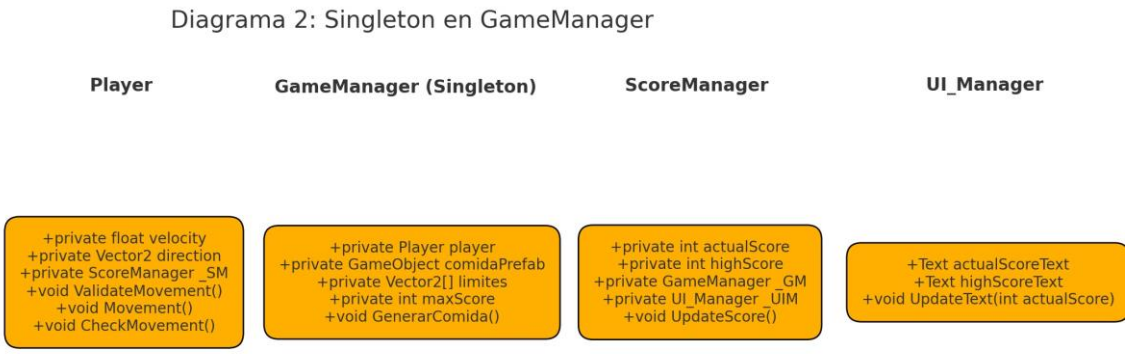

Diagrama 2

Consideré el GameManager como Singleton porque es una de las mejores opciones en muchos juegos, incluido Snake. El GameManager coordina todas las mecánicas principales del juego, como los estados de inicio y fin, la generación de comida, y la administración de las reglas. Al asegurarme de que solo exista una instancia del

GameManager, se evita que haya conflictos y duplicaciones innecesarias en el control del flujo del juego.

La ventaja de esta opción es que me permite manejar toda la lógica del juego desde un solo lugar, evitando problemas como duplicación de instancias o desincronización. Al usar un Singleton, los eventos importantes como la generación de comida o la administración de los estados del juego (pausa, reinicio, etc.) son coherentes y están bien gestionados.

Sin embargo, en el prototipo decidí no usar el GameManager como Singleton, ya que el control principal del juego se maneja de manera más directa y centralizada a través del Player.

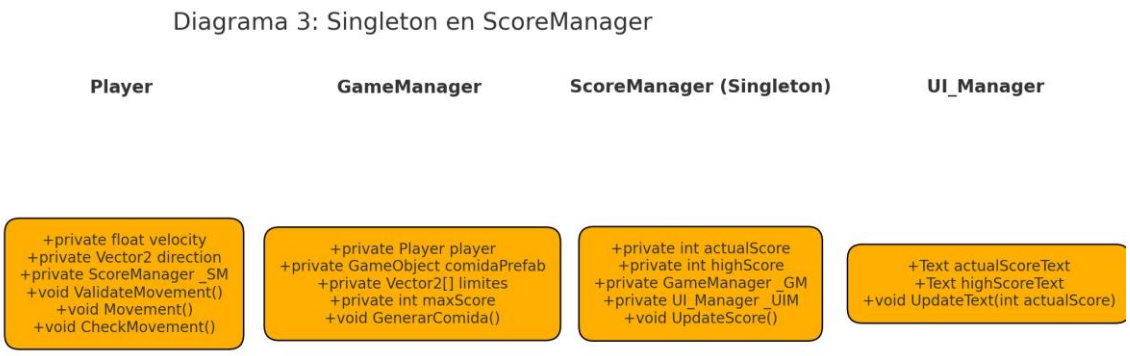

Diagrama 3

También consideré hacer del ScoreManager un Singleton porque es responsable de manejar las puntuaciones, un aspecto clave en un juego como Snake. Con un Singleton, se garantizaría que las puntuaciones sean consistentes y siempre accesibles desde cualquier parte del código, sin necesidad de referencias complicadas. Esto es importante porque las puntuaciones deben mantenerse seguras y no reiniciarse accidentalmente.

Aun así, decidí que el ScoreManager no necesitaría ser Singleton en esta fase del prototipo, ya que el control principal y la información más relevante están centralizados en el Player, quien maneja la interacción y el progreso en el juego.